08

Mãos à obra: alteração de livros

Chegou a hora de consolidar a funcionalidade de alteração do nosso CRUD que você aprendeu no capítulo. Seguem linhas gerais para guiá-lo nesta tarefa, mas claro, sem perder o teor de desafio.

1 - O primeiro passo é sabermos qual da nossa lista queremos alterar. Adicione mais uma coluna na tabela de livros que chamará o método `carregar` que recebe o livro que queremos alterar:

```
<!-- antes da coluna da funcionalidade de remoção -->
        <h:column>
            <f:facet name="header">Alterar</f:facet>
            <h:commandLink value="altera" action="#{livroBean.carregar(livro)}" />
        </h:column>
```

2 - E claro, o método `carregar` de `LivroBean` :

```
public void carregar(Livro livro) {
    System.out.println("Carregando livro " + livro.getTitulo());
    this.livro = livro;
}
```

3 - Você lembra qual problema teremos se deixarmos a aplicação do jeito que está? Teremos um `LazyInitializationException` . A primeira forma de resolver é alterar o relacionamento entre `Livro` e `Autor` como EAGER.

4 - Isso ainda não é suficiente. Como nosso método `gravar` será capaz de distinguir uma alteração de uma inclusão? Um Livro sem ID é aquele que nunca foi incluído no banco, já Livros que possuem ID são aqueles que já foram incluídos e estão sendo recuperados para alguma outra finalidade, por exemplo, consulta ou alteração. Por isso vamos colocar uma condição `if` em nosso método `gravar` :

```
public void gravar() {
    System.out.println("Gravando livro " + this.livro.getTitulo());

    if (livro.getAutores().isEmpty()) {
        FacesContext.getCurrentInstance().addMessage("autor",
                new FacesMessage("Livro deve ter pelo menos um Autor."));
        return;
    }

    if (this.livro.getId() == null) {
        new DAO<Livro>(Livro.class).adiciona(this.livro);
    } else {
        new DAO<Livro>(Livro.class).atualiza(this.livro);
    }

    this.livro = new Livro();
}
```

5 - Agora é só testar e verificar o resultado.